

BasicCard

C-API

Version: 4.30.00

Date: 05.03.02 12:31

Author: Michael Petig

email: development@zeitcontrol.de

Web sites:

<http://www.zeitcontrol.de>

<http://www.basiccard.com>

<http://www.basiccardusa.com>

Contents

BasicCard	1
C-API	1
1. Overview	1
1.1 Design overview	1
1.1.1 Reader Wrapper Layer ZCRWL.....	1
1.1.2 Common Reader Interface ZCCRI	1
1.1.3 BasicCard Command Interface ZCBCI	2
1.2 Tools.....	2
1.2.1 TestCri.....	2
1.2.2 ZCCriReg	2
1.2.3 DefReg.....	2
2. Developers Guide	3
2.1 Setup Build Environment for C/C++ Compilers	3
2.2 Setup Runtime Environment.....	3
2.3 Redistribute Runtime Environment	3
2.4 Programming Intro	4
2.4.1 Header Files.....	4
2.4.2 Using ZCCRI.....	4
2.4.3 Using ZCBCI.....	5
2.4.4 Call User Defined Card Command by Use of ZCBCI.....	5
3. Reference.....	7
3.1 ZCCRI	7
3.1.1 Functions by Category.....	7
3.1.2 Functions by Name.....	8
3.1.3 Structures and Types	17
3.1.4 Constants	19
3.2 ZCBCI	20
3.2.1 Functions by Category.....	20
3.2.2 Functions by Name.....	21
3.2.3 Structures and Types	52
3.2.4 Constants	57
4. Index.....	61

1. Overview

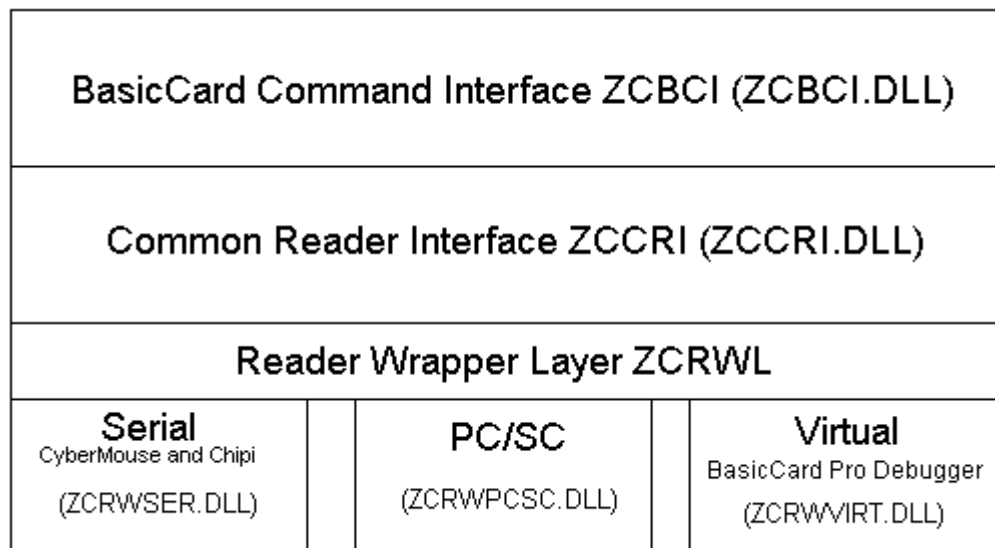
This document describes the BasicCard-C-API. This API enables developers to use both operating system features like GUI support for example, and BasicCard support to create BasicCard applications almost as simply as if using ZCBasic. This API is currently available for Windows™ 32 bit operating systems. The API supports the ZeitControl CyberMouse card reader, ZeitControl Chipi card reader and PC/SC compatible readers on these operating systems. It is possible to plug in different readers by creating a small support library. Further the API can use the BasicCard Professional Card Debugger as virtual card reader.

We decided to develop a C-API first, since this kind of API can be used from almost all programming languages and tools, while C++ libraries cannot be shared and used by different compilers. Since it is difficult to use from Visual Basic™ we have developed an ActiveX control (OCX) which is included with this package. For now we have tested and optimized it for use by Visual Basic™ 6.0. For reference how to use this OCX please refer to documentation include for this purpose. Anyway it could be a good choice to read this chapter even if using Visual Basic™ and not C.

Note: To understand this document it is strongly recommended to read the BasicCard manual first.

1.1 Design overview

This API consists of several DLLs, which are placed in one of 3 layers.



Picture 1

1.1.1 Reader Wrapper Layer ZCRWL

This layer consists of one DLL for each supported reader. Normally BasicCard developers will never need to use these libraries directly. But if you want to plug in your own reader, you should take a look at RWL SDK, included in the distributed C-API package. There is a template included, which may be extended.

1.1.2 Common Reader Interface ZCCRI

This layer is needed to create a connection to one of the supported readers. Any reader can be opened by giving a name consisting of the service name (“PCSC”, “Serial” or “Debug”) and the readers native name (for example: “COM1” for using a serial CyberMouse reader using serial service DLL ZCRWSER.DLL), separated by a colon (:). Number and name of the supported readers can be queried. Supported readers are detected at runtime by checking the registration info about installed reader plugins (ZCRWL-DLLs), loading registered plugins and asking them for known readers. When a reader is opened all further requests are forwarded to the related plugin DLL.

ZCCRI includes functions to open and close a reader device, wait for a card to be inserted or query if card is inserted, connect to card, reconnect (reset) card or disconnect from card. In addition to this, simple transaction functions are provided to be used by ZCBCI. Normally developers will not need to call these transaction

1. Overview

commands directly. To support reader plugins, functions to register and deregister RWL services are provided. Since it is possible to open a default reader there are additional functions to set and query this default reader. For easy use a simple dialog to select one of all supported readers is also included.

1.1.3 BasicCard Command Interface ZCBCI

This is the major part of the API. This layer uses the ZCCRI layer to communicate with the card and the reader. So to use this you must first use the ZCCRI to create a connection to the card and reader, then attach this reader device to your card object.

ZCBCI includes almost all predefined commands (including full file system support for Enhanced BasicCard) of Enhanced and Compact BasicCard. Further a simple transaction command enables you to create access function to user defined card commands. As in ZCBasic transparent encryption is included in ZCBCI. To enable this, additional functions to load and set keys and polynomials are available. To connect to the reader device, functions to attach to and detach from the device (ZCCRI) are provided.

1.2 Tools

1.2.1 TestCri

TestCri.exe enables you to quickly check your installation. All you need is a supported card reader and either a Compact or Enhanced BasicCard.

1.2.2 ZCCriReg

ZCCriReg.exe could be used to register and deregister RWL service DLLs and to set default reader.

1.2.3 DefReg

This simple batch file (defreg.bat) completes your installation by registering the Chipi and the PCSC RWL service and ask for your default reader.

Note: For performance reasons you may want to deregister services not used by ZCCriReg later.

Note: With new RWL-DLLs you may also use Windows RegSvr32.exe utility to register and deregister the service DLL.

2. Developers Guide

2.1 Setup Build Environment for C/C++ Compilers

Copy all header files (*.h) from install C-API H directory (default: "C:\BasicCardPro\API\H") to your include directory, your project directory or extend the INCLUDE environment variable to include C-API H directory. Copy all library files (*.lib) to your library directory, your project directory or extend the LIB environment variable to include C-API Lib directory (default: "C:\BasicCardPro\API\Lib"). Include "zcbci.h" in all source files which use ZCCRI or ZCBCI functions. Link your executable and DLLs with "zccri.lib" and "zcbci.lib". For details on how to apply these settings to your development environment refer to your C/C++ compiler reference. To make libraries available for as many development systems as possible we have created libraries for several environments. This is necessary because most times libraries created for one compiler/linker will not work with a different set of compiler linker. So our install package includes libraries in following formats:

- Microsoft™ Visual C/C++ 5.0 and others (default: "C:\BasicCardPro\API\Lib")
- Microsoft™ new library format (default: "C:\BasicCardPro\API\Lib\MS_NEW")
- Borland™ C-Builder (default: "C:\BasicCardPro\API\Lib\Borland")

If you still run into problems using import libraries "zccri.lib" and "zcbci.lib", you may use DLLs "zccri.dll" and "zcbci.dll" or def files "zccri.def" and "zcbci.def" to create your own import libraries.

Note: If you have used another release version before you should use new header and library files but you do not need to recompile old programs.

2.2 Setup Runtime Environment

After finishing the installation of the BasicCard API the required files are copied to your system directory and all supplied RWL service DLLs are automatically registered. You may want to deregister unused RWL service DLLs for performance reasons. The BasicCard API runtime currently consists of following DLLs:

- ZCBCI.DLL
- ZCCRI.DLL
- ZCRWSER.DLL
- ZCRWPCSC.DLL requires PC/SC support to be installed
- ZCPLIB.DLL (runtime DLL used by other DLLs)

Additionally the following tools are included:

- ZCCRIREG.EXE
- TESTCRI.EXE
- DEFREG.BAT

Additionally the following DLL is used for development, but not required for use by your customer:

- ZCRWVIRT.DLL (virtual card reader for BasicCard Pro Card Debugger)

2.3 Redistribute Runtime Environment

We have decided to make it as easy as possible for you to include necessary files in your own setup programs. For this purpose we have decided to support new "Windows Installer" API. Take a look at http://www.installsite.org/w2k_msiauth.htm for products supporting this API. Following files (merge modules) may be used to be included in your own setup routines:

- ZCBCI.MSM
- ZCCRI.MSM
- ZCRWSER.MSM (required when using ZeitControl CyberMouse or Chipi serial card reader without PC/SC)
- ZCRWPCSC.MSM (required when using other PC/SC card readers)
- ZCSLRHLP.MSM (this will allow your customers to choose a default reader at end of installation)

For customers not using a Windows Installer compatible installer, a ready to run runtime installation package is available. You can find this on the software CD or at <http://www.basiccard.com>.

2.4 Programming Intro

2.4.1 Header Files

You may find additional information by browsing through the header files.

- ZCBCI.H constants, typedefs and function prototypes of ZCBCI
- ZCCRI.H global ZCCRI header file includes ZCCRIDEF.H, ZCCRITYP.H and ZCCRIPTH.H
- ZCCRIDEF.H constants used by ZCCRI
- ZCCRITYP.H typedefs used by ZCCRI
- ZCCRIPTH.H function prototypes of all ZCCRI functions

2.4.2 Using ZCCRI

Before you can call any BasicCard command you must first open a reader device and establish a connection to the card in reader. So following examples shows how to do this:

```
/* Variables used */
ZCCRIRET criret; /* ZCCRI return codes */
ZCCRIREADER reader; /* handle to ZCCRI reader device */
/* If no check for connected readers is required we */
/*can startup much more fast by use of ZCRI_OPTION_FASTINIT.*/
criret=ZCCriSetOptions2(NULL, ZCCRI_OPTION_FASTINIT);
if (ZCCRI_NOERROR!=criret)
{
    /* Do error handling here */
    return;
}
/* First open reader device, here use default reader (NULL) */
criret=ZCCriOpenReader(&reader, NULL);
if (ZCCRI_NOERROR!=criret)
{
    /* Do error handling here */
    return;
}
/* Then wait for card, here wait indefinite (-1) */
/* When using this with indefinite wait keep in mind,*/
/* this will block the message queue. So instead we */
/* recommend to use ZCCriWaitCard with wait 0 in */
/* conjunctions with a Windows timer*/
criret=ZCCriWaitCard(reader, -1);
switch (criret)
{
    case ZCCRI_NOERROR:
        /* there is a card in reader */
        break;
    case ZCCRI_ERROR_NO_CARD:
        /* no card in reader, should not happen if indefinite wait */
        return;
    default:
        /* do error handling here */
        return;
}
/* Connect to inserted card, here without getting card ATR (NULL) */
criret=ZCCriConnect(reader, NULL);
if (ZCCRI_NOERROR!=criret)
{
```

```

        /* Do error handling here */
        return;
    }
    /* Do your BasicCard work here. In this example call function which */
    /* will be discussed later */
    DoBasicJob(reader);
    /* Now cleanup by disconnect from card */
    criret=ZCCriDisconnect(reader);
    if (ZCCRI_NOERROR!=criret)
    {
        /* Do error handling here */
        return;
    }
    /* Then close reader */
    criret=ZCCriCloseReader(reader);

```

2.4.3 Using ZCBCI

After you have established a valid connection to your card you can start doing you BasicCard commands. But first you must attach to the reader.

Note: You should never attach more than one ZCBCI context (ZCBCICARD) to one reader context at the same time.

```

void DoBasicJob(ZCCRIREADER reader)
{
    ZCBCIRET bciret; /* ZCBCI return codes */
    WORD sw; /* status word returned by ZCBCI commands */
    ZCBCICARD card; /* handle to BasicCard */
    char echo[]="ABC"; /* string used by echo command */
    /* attach to reader */
    bciret=ZCBciAttach(reader, &card);
    if (ZCBCI_NOERROR!=bciret)
    {
        /* Do error handling here */
        return;
    }
    /* Call BasicCard command, here standard echo command with */
    /* increment 1. This should return "BCD" into echo buffer */
    bciret=ZCBciEcho(card, &sw, 1, echo);
    if (ZCBCI_NOERROR!=bciret)
    {
        /* Do error handling here */
        return;
    }
    /* Call other commands here. You may also use ZCCriReconnect */
    /* to reset BasicCard. */
    /* Than cleanup by detach from reader. */
    bciret=ZCBciDetach(card);
    if (ZCBCI_NOERROR!=bciret)
    {
        /* Do error handling here */
        return;
    }
}

```

2.4.4 Call User Defined Card Command by Use of ZCBCI

To use user defined commands with ZCBCI you should create a small wrapper function for each command. If for example there is a card command which is defined in ZCBasic as follows

Declare Command &H20 &H08 MyCommand (bVal as Byte, iVal as Integer, lVal as Long, strVal as String)

your wrapper function may look like this:

2. Developers Guide

```
ZCBCIRET MyCommand (unsigned char *pByte, short *pInteger, long *pLong,
char *pszString, int cbString)
{
    ZCBCIRET bciret; /* ZCBCI return codes */
    ZCBCIPARM parm[4]; /* 4 parameters required for command */
    WORD sw; /* status word returned by transaction */

    /* setup parameters from left to right as declared for command */
    parm[0].parmtype=ZCBCIBYTE; /* first parameter is of type byte */
    parm[0].data.valbyte=*pByte; /* value is *pByte */
    parm[1].parmtype=ZCBCIINT; /* second parameter is of type ZCBasic */
    parm[1].data.valint=*pInteger; /* integer (16 bit). value is */
    /* *pInteger */
    parm[2].parmtype=ZCBCILONG; /* third parameter is of type long */
    parm[2].data.vallong=*pLong; /* value is *pLong */
    parm[3].parmtype=ZCBCISTRING; /* fourth parameter is of type string*/
    parm[3].data.valstring.cbString=(BYTE) min(255, cbString);
    /* maximum length of string buffer is*/
    /* cbString. Since target type is */
    /* BYTE it should not exceed byte */
    /* range (0..255) */
    parm[3].data.valstring.pString=pszString; /* value is pszString */

    /* call transaction with Cla 0x20 (&H20) and Ins 0x08 (&H08) */
    /* give parameter and parameter count (4) */
    bciret=ZCBCiTransaction(card, 0x20, 0x08, parm, 4, &sw);
    if (ZCBCI_NOERROR!=bciret)
    {
        /* Do error handling here */
        /* Normally it is not required to check sw, since bad sw */
        /* will cause bciret to be ZCBCI_ERROR_TRANS_FAILED */
        return;
    }

    /* if desired copy returned parameter back to function parameters */
    *pByte=parm[0].data.valbyte;
    *pInteger=parm[1].data.valint;
    *pLong=parm[2].data.vallong;
    /* string parameter is not copied back, since this has already */
    /* happened because of pointer usage */

    /* return to caller with return value */
    Return bciret;
}
```

3. Reference

3.1 ZCCRI

3.1.1 Functions by Category

3.1.1.1 Management Functions

- ZCCriSetDefaultReader
- ZCCriGetDefaultReader
- ZCCriRegisterService
- ZCCriDeregisterService

3.1.1.2 Name Services

- ZCCriGetReaderCount
- ZCCriGetReaderName
- ZCCriSelectReaderDialog

3.1.1.3 Reader Access Services

- ZCCriOpenReader
- ZCCriCloseReader

3.1.1.4 Connection Services

- ZCCriWaitCard
- ZCCriConnect
- ZCCriReconnect
- ZCCriDisconnect

3.1.1.5 Low Level Transaction Functions

- ZCCriTransaction
- ZCCriTransaction2

3.1.1.6 Debug Support Functions

- ZCCriSetLogFile

3.1.1.7 Extended Reader Functions

- ZCCriCardInReader
- ZCCriGetFeatures
- ZCCriGetOptions
- ZCCriGetOptions2
- ZCCriSetOptions
- ZCCriSetOptions2

3. Reference

3.1.2 Functions by Name

3.1.2.1 ZCCriCardInReader

Check if card is inserted.

Note: This function requires a RWL service supporting this function. If unsupported (as in most cases) ZCCriWaitCard should be used. If supported ZCCRI_FEATURE_CHECKCARD will be returned by ZCCriGetFeatures. If supported with returning ATR ZCCRI_FEATURE_CHECKCARD_ATR will also be returned.

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriCardInReader(PZCCRIREADERNAME pName, PZCCRIATR pAtr);
```

Parameter:

Number	Name	Direction	Description
1	pName	in	Name of reader to check
2	*pAtr	out	ATR of inserted card (if supported)

Return values:

Value	Description
ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_NO_CARD	No error, no card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown
ZCCRI_ERROR_NOT_SUPPORTED	This call is not supported by this reader or service

See also: ZCCriWaitCard

3.1.2.2 ZCCriCloseReader

Close reader device.

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriCloseReader (ZCCRIREADER hReader);
```

Parameter:

Number	Name	Direction	Description
1	hReader	in	Handle to reader device

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCCriOpenReader

3.1.2.3 ZCCriConnect

Establish connection to card in given reader device.

Syntax:

ZCCRIRET ZCCRILINK ZCCriConnect (ZCCRIREADER hReader, PZCCRIATR pAtr);

Parameter:

Number	Name	Direction	Description
1	hReader	in	Handle to reader device
2	*pAtr	out	Card ATR. If pAtr is set to NULL no ATR is returned

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKNOWN_READER	Reader not known
ZCCRI_ERROR_READER_BUSY	Reader busy
ZCCRI_ERROR_NO_CARD	No card in reader

See also: ZCCriReconnect, ZCCriDisconnect

3.1.2.4 ZCCriDeregisterService

Deregister RWL service DLL.

Syntax:

ZCCRIRET ZCCRILINK ZCCriDeregisterService (CHAR *pszService);

Parameter:

Number	Name	Direction	Description
1	*pszService	in	Name of service to deregister

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_MISC	Unknown system error

See also: ZCCriDeregisterService

3.1.2.5 ZCCriDisconnect

Disconnect from card in reader device.

Syntax:

ZCCRIRET ZCCRILINK ZCCriDisconnect (ZCCRIREADER hReader);

Parameter:

3. Reference

Number	Name	Direction	Description
1	hReader	In	Handle to reader device

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCCriConnect, ZCCriReconnect

3.1.2.6 ZCCriGetDefaultReader

Get default reader used when NULL or empty String is used in ZCCriOpenReader call.

Syntax:

ZCCRIRET ZCCRILINK ZCCriGetDefaultReader (PZCCRIREADERNAME pName);

Parameter:

Number	Name	Direction	Description
1	*pName	Out	Name of default reader.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_NO_DEFAULT	No default reader defined

See also: ZCCriSetDefaultReader, ZCCriSelectReaderDialog

3.1.2.7 ZCCriGetFeatures

Query extended features supported by reader or service.

Syntax:

ZCCRIRET ZCCRILINK ZCCriGetFeatures(PZCCRIREADERNAME pName, DWORD *pdwFeatures);

Parameter:

Number	Name	Direction	Description
1	pName	in	Name of reader, service (e.g. "PCSC:") or NULL for default reader to query.
2	*pdwFeatures	out	Supported features as bitmap of ZCCRI_FEATURE_...

Return values:

Value	Description
ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown

See also:

3.1.2.8 ZCCriGetOptions

Get option of open card reader.

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriGetOptions(ZCCRIREADER hReader, DWORD *pdwOptions);
```

Parameter:

Number	Name	Direction	Description
1	hReader	in	Handle of open card reader.
2	*pdwOptions	out	Device option as bitmap of ZCCRI_OPTION_...

Return values:

Value	Description
ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown
ZCCRI_ERROR_NOT_SUPPORTED	Not supported by this service.

See also: ZCCriGetOptions2

3.1.2.9 ZCCriGetOptions2

Get option of for single service (RWL DLL).

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriGetOptions2(PZCCRISERVICENAME pServiceName, DWORD *pdwOptions);
```

Parameter:

Number	Name	Direction	Description
1	pServiceName	in	Name of service.
2	*pdwOptions	out	Device option as bitmap of ZCCRI_OPTION_...

Return values:

Value	Description
ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown
ZCCRI_ERROR_NOT_SUPPORTED	Not supported by this service.

See also: ZCCriGetOptions

3. Reference

3.1.2.10 ZCCriGetReaderCount

Get number of supported reader devices.

Syntax:

ZCCRIRET ZCCRILINK ZCCriGetReaderCount (PULONG pCount);

Parameter:

Number	Name	Direction	Description
1	*pCount	out	Number of supported reader devices

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCCriGetReaderName, ZCCriSelectReaderDialog

3.1.2.11 ZCCriGetReaderName

Get name of supported reader addressed by number ulNum.

Syntax:

ZCCRIRET ZCCRILINK ZCCriGetReaderName (ULONG ulNum, PZCCRIREADERNAME pName);

Parameter:

Number	Name	Direction	Description
1	ulNum	in	Number of requested reader (0 .. readercount-1)
2	*pName	out	Name of requested reader

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader not known (ulNum >= readercount)

See also: ZCCriGetReaderCount, ZCCriSelectReaderDialog

3.1.2.12 ZCCriOpenReader

Open reader device

Syntax:

ZCCRIRET ZCCRILINK ZCCriOpenReader (PZCCRIREADER phReader, PZCCRIREADERNAME pName);

Parameter:

Number	Name	Direction	Description
1	*phReader	out	Handle to reader device
2	*pName	in	Name of reader device to open. NULL for default reader.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader not known or not supported
ZCCRI_ERROR_READER_BUSY	Reader busy or not accessible

See also: ZCCriCloseReader, ZCCriGetReaderName, ZCCriSelectReaderDialog, ZCCriSetDefaultReader

3.1.2.13 ZCCriReconnect

Reconnect to card in reader or in other words reset card.

Syntax:

ZCCRIRET ZCCRILINK ZCCriReconnect (ZCCRIREADER hReader, PZCCRIATR pAtr);

Parameter:

Number	Name	Direction	Description
1	hReader	in	Handle to reader device
2	*pAtr	out	Card ATR. If pAtr is set to NULL no ATR is returned

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_NO_CARD	No card in reader

See also: ZCCriConnect, ZCCriDisconnect

3.1.2.14 ZCCriRegisterService

Register a RWL service DLL.

Syntax:

ZCCRIRET ZCCRILINK ZCCriRegisterService (CHAR *pszService, CHAR *pszDLLName);

Parameter:

Number	Name	Direction	Description
1	*pszService	In	Name of service to register (must be the same as used in RWL DLL).
2	*pszDLLName	In	Name of DLL implements this service

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_MISC	Unknown system error

3. Reference

See also: ZCCriDeregisterService

3.1.2.15 ZCCriSelectReaderDialog

Show dialog to select a reader device.

Syntax:

ZCCRIRET ZCCRILINK ZCCriSelectReaderDialog (PZCCRIREADERNAME pName);

Parameter:

Number	Name	Direction	Description
1	*pName	Out	Name of selected reader device to open.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_CANCEL	Dialog cancelled, no reader selected

See also: **Fehler! Verweisquelle konnte nicht gefunden werden.**, ZCCriGetReaderName

3.1.2.16 ZCCriSetDefaultReader

Set default reader used when NULL or empty String is used in ZCCriOpenReader call.

Syntax:

ZCCRIRET ZCCRILINK ZCCriSetDefaultReader (PZCCRIREADERNAME pName);

Parameter:

Number	Name	Direction	Description
1	*pName	In	Name of default reader.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_MISC	Unknown system error

See also: ZCCriGetDefaultReader, **Fehler! Verweisquelle konnte nicht gefunden werden.**

3.1.2.17 ZCCriSetLogFile

Log all input and output data to log file.

Syntax:

ZCCRIRET ZCCRILINK ZCCriSetLogFile (ZCCRIREADER hReader, CHAR *pszFile);

Parameter:

Number	Name	Direction	Description
--------	------	-----------	-------------

1	hReader	In	Handle to reader device
2	*pszFile	In	Name of log file,. NULL to disable logging

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_MISC	Unable to open file

3.1.2.18 ZCCriSetOptions

Set options of open card reader.

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriSetOptions(ZCCRIREADER hReader, DWORD dwOptions);
```

Parameter:

Number	Name	Direction	Description
1	hReader	in	Handle of open card reader.
2	dwOptions	in	Device option to set as bitmap of ZCCRI_OPTION_...

Return values:

Value	Description
ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown
ZCCRI_ERROR_NOT_SUPPORTED	Not supported by this service.

See also: ZCCriSetOptions2

3.1.2.19 ZCCriSetOptions2

Set option of for single service (RWL DLL). Especially useful for make CRI initialization more fast by use of ZCCRI_OPTION_FASTINIT. This is recommended if check for really connected readers is not required.

Note: When use for ZCCRI_OPTION_FASTINIT this must be first call to ZCCri... functions and pServiceName must be NULL.

Syntax:

```
ZCCRIRET ZCCRILINK ZCCriSetOptions2(PZCCRISERVICENAME pServiceName, DWORD dwOptions);
```

Parameter:

Number	Name	Direction	Description
1	pServiceName	in	Name of service or NULL for all services.
2	dwOptions	in	Device option as bitmap of ZCCRI_OPTION_...

Return values:

Value	Description
-------	-------------

3. Reference

ZCCRI_NOERROR	No error, card is inserted
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_UNKOWN_READER	Reader or service specified by pName is unknown
ZCCRI_ERROR_NOT_SUPPORTED	Not supported by this service.

See also: ZCCriSetOptions

3.1.2.20 ZCCriTransaction

Send (and receive) T1 transaction to (from) smart card.

Note: This function is for low level use as by ZCBCI-API. Use ZCBciTransaction instead of this function, whenever possible.

ZCCRIRET ZCCRILINK ZCCriTransaction (ZCCRIREADER hReader, BYTE *pDataIn, DWORD cbDataIn, BYTE *pReply, DWORD *pcbReply);

Parameter:

Number	Name	Direction	Description
1	hReader	In	Handle to reader device
2	*pDataIn	In	Data to sent to smart card
3	cbDataIn	In	Number of bytes send to smart card
4	*pReply	Out	Data received from smart card
5	*pcbReply	In	Maximum number of bytes to receive (size of receive buffer *pReply)
		Out	Number of bytes received

Return values:

ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_PROTOCOL	Protocol (T1) error transaction failed
ZCCRI_ERROR_NO_CARD	No card in reader, card removed
ZCCRI_ERROR_OVERFLOW	Reply buffer overflow, buffer too small

See also: ZCCriTransaction2

3.1.2.21 ZCCriTransaction2

Send (and receive) T1 transaction to (from) smart card.

Note: This function is for low level use as by ZCBCI-API. Use ZCBciTransaction instead of this function, whenever possible.

Syntax:

ZCCRIRET ZCCRILINK ZCCriTransaction2 (ZCCRIREADER hReader, BYTE Cla, BYTE Ins, BYTE P1, BYTE P2, BYTE Lc, BYTE *pData, SHORT Le, BYTE *pReply, SHORT *pcbReply, WORD *pSW1SW2);

Parameter:

Number	Name	Direction	Description
1	hReader	In	Handle to reader device

2	Cla	In	Class byte
3	Ins	In	Instruction byte
4	P1	In	Parameter 1
5	P2	In	Parameter 2
6	Lc	In	Lc byte
7	*pData	In	Idata send to smart card
8	Le	In	Le byte
9	*pReply	Out	Data received from smart card (without SW1, SW2)
10	*pcbReply	In	Size of reply buffer
		Out	Number of bytes received
11	*pSW1SW2	Out	Command status word SW1, SW2

Return values:

ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_PROTOCOL	Protocol (T1) error transaction failed
ZCCRI_ERROR_NO_CARD	No card in reader, card removed
ZCCRI_ERROR_OVERFLOW	Reply buffer overflow, buffer too small

3.1.2.22 ZCCriWaitCard

Wait for card to be inserted.

Syntax:

ZCCRIRET ZCCRILINK ZCCriWaitCard (ZCCRIREADER hReader, LONG ITimeout);

Parameter:

Number	Name	Direction	Description
1	hReader	In	Handle to reader device
2	ITimeout	In	Maximum time to wait (ms). Use -1 for indefinite wait.

Return values:

ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCCRI_ERROR_NO_CARD	No card in reader, card removed
ZCCRI_ERROR_READER_BUSY	Card reader busy

3.1.3 Structures and Types

Pointer types to defined types can be constructed by prefix type with 'P'. For example PZCCRIRET is a pointer to ZCCRIRET. All types are aligned on byte boundaries (#pragma pack(1)).

3. Reference

3.1.3.1 ZCCRIRET, PZCCRIRET

32 bit unsigned ZCCRI-API return value. See Table 1 ZCCRI return values for possible values.

C typedef:

```
typedef DWORD ZCCRIRET;  
typedef ZCCRIRET *PZCCRIRET;
```

3.1.3.2 ZCCRIVER, PZCCRIVER

16 bit ZCCRI-API version number.

C typedef:

```
typedef WORD ZCCRIVER;  
typedef ZCCRIVER *PZCCRIVER;
```

3.1.3.3 ZCCRHANDLE, PZCCRHANDLE

32 bit handle.

C typedef:

```
typedef DWORD ZCCRHANDLE;  
typedef ZCCRHANDLE *PZCCRHANDLE;
```

3.1.3.4 ZCCRIREADER, PZCCRIREADER

Handle to open reader device.

C typedef:

```
typedef ZCCRHANDLE ZCCRIREADER;  
typedef ZCCRIREADER *PZCCRIREADER;
```

3.1.3.5 ZCCRISERVICENAME, PZCCRISERVICENAME

ZCCRI service name. For example “Chipi” or “PCSC”.

C typedef:

```
typedef CHAR ZCCRISERVICENAME[ZCCRI_MAX_SERVICE_NAME+1];  
typedef CHAR *PZCCRISERVICENAME;
```

3.1.3.6 ZCCRIREADERNAME, PZCCRIREADERNAME

ZCCRI reader name. For example “Chipi:COM1” or “PCSC:ZEITCONTROL Chipi 0”.

C typedef:

```
typedef CHAR ZCCRIREADERNAME[ZCCRI_MAX_READER_NAME+1];  
typedef CHAR *PZCCRIREADERNAME;
```

3.1.3.7 ZCCRIATR, PZCCRIATR

Type to hold card ATR (Answer To Reset). Consists of length field (bLength) and byte array to hold ATR (abAtr).

C typedef:

```
typedef struct _ZCCRIATR  
{  
    BYTE bLength;
```

```

    BYTE abAtr[ZCCRI_MAX_ATR+1];
} ZCCRIATR;
typedef ZCCRIATR *PZCCRIATR;

```

3.1.4 Constants

Value (Hex)	Symbolic name	Description
0	ZCCRI_NOERROR	Function call successful, no error
0x01000000	ZCCRI_ERROR_PARM	Invalid parameter passed to function call
0x01000001	ZCCRI_ERROR_UNKOWN_READER	Given reader is not known or supported
0x01000002	ZCCRI_ERROR_READER_BUSY	Reader is busy
0x01000003	ZCCRI_ERROR_PROTOCOL	Transport protocol (T=1) error
0x01000004	ZCCRI_ERROR_MEM	Out of memory
0x01000005	ZCCRI_ERROR_INITFAILED	Initialization failed
0x01000006	ZCCRI_ERROR_MISC	Misc. error
0x01000007	ZCCRI_ERROR_NO_CARD	No card in reader, card removed
0x01000008	ZCCRI_ERROR_NO_DEFAULT	No default choice (reader) available or defined
0x01000009	ZCCRI_CANCEL	Action canceled (by user)
0x0100000A	ZCCRI_ERROR_OVERFLOW	Buffer overflow, buffer too small to hold output data
0x0100000B	ZCCRI_ERROR_NOT_SUPPORTED	This function is not supported by this reader or service.

Table 1 ZCCRI return values

Symbolic name	Value	Description
ZCCRILINK	APIENTRY	ZCCRI-API linkage convention
ZCCRI_MAX_SERVICE_NAME	16	Max length of service names
ZCCRI_MAX_READER_NAME	256	Max length of ZCCRI reader names
ZCCRI_MAX_ATR	34	Max length of ATR
ZCCRI_SEPARATOR	“.”	Separator used between service name and native reader name. For example if service name is “PCSC” and native reader name is “ZEITCONTROL Chipi 0” this results in full reader name “PCSC:ZEITCONTROL chipi 0”
ZCCRI_FEATURE_CHECKCARD	1	Use with ZCCriGetFeatures. If returned ZCCriCardInReader is supported
ZCCRI_FEATURE_CHECKCARD_ATR	2	Use with ZCCriGetFeatures. If returned ZCCriCardInReader with ATR is supported
ZCCRI_OPTION_FASTINIT	1	Use with ZCCriSetOptions2 to make CRI initialization more fast

Table 2 Other ZCCRI constants

3.2 ZBCBI

3.2.1 Functions by Category

3.2.1.1 Attachment Functions

- ZCBciAttach
- ZCBciDetach

3.2.1.2 Key Management Functions

- ZCBciReadKeyFile
- ZCBciSetKey
- ZCBciSetPoly

3.2.1.3 Transaction Functions

- ZCBciTransaction
- ZCBciTransaction2

3.2.1.4 Eeprom Access Functions

- ZCBciEepromCRC
- ZCBciEepromSize
- ZCBciClearEeprom
- ZCBciReadEeprom
- ZCBciWriteEeprom

3.2.1.5 Encryption Functions

- ZCBciStartEncryption
- ZCBciEndEncryption

3.2.1.6 Card Management Functions

- ZCBciGetApplicationID
- ZCBciEcho
- ZCBciGetState
- ZCBciSetState

3.2.1.7 File System Functions

- ZCBciChDir
- ZCBciCloseAllFiles
- ZCBciCloseFile
- ZCBciCurDir
- ZCBciDirCount
- ZCBciDirFile
- ZCBciEraseFile

- ZCBciFileGet
- ZCBciFileGetPos
- ZCBciFileLength
- ZCBciFilePrint
- ZCBciFilePut
- ZCBciFilePutPos
- ZCBciFileRead
- ZCBciFileReadLine
- ZCBciFileReadSingle
- ZCBciFileReadString
- ZCBciFileReadUser
- ZCBciFileReadUser2
- ZCBciFileWrite
- ZCBciFileWriteLong
- ZCBciFileWriteSingle
- ZCBciFileWriteString
- ZCBciFileWriteUser
- ZCBciGetAttr
- ZCBciGetFilepos
- ZCBciGetLockInfo
- ZCBciMkDir
- ZCBciOpenFile
- ZCBciQueryEof
- ZCBciRename
- ZCBciRmdir
- ZCBciSetFileLock
- ZCBciSetFilepos

3.2.2 Functions by Name

3.2.2.1 ZCBciAttach

Attach reader device to ZCBCI card handle. Must be first call before card handle can be passed to any other function.

Syntax:

ZCBCIRET ZCBCILINK ZCBciAttach (ZCCRIREADER reader, PZCBCICARD pCard);

Parameter:

Number	Name	Direction	Description
1	Reader	in	Handle to reader device
2	*pCard	out	Handle to BasicCard

3. Reference

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCCriOpenReader, ZCBciDetach

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.2 ZCBciChDir

Change current directory on card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciChDir (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	New active directory to set.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciDirCount, ZCBciDirFile

Supported cards: ZC2.X, ZC3.X

3.2.2.3 ZCBciClearEeprom

BasicCard CLEAR EEPROM command (Cla C0, Ins 04). Clear (set to FF) length bytes of eeprom starting at address start.

Note: Keep in mind that CLEAR EEPROM will not send WTX, so do not clear more than 1 kB of eeprom memory with one call.

Syntax:

ZCBCIRET ZCBCILINK ZCBciClearEeprom (ZCBCICARD card, WORD *pSW1SW2, WORD start, WORD length);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card

2	*pSW1SW2	Out	Command status word
3	start	In	start address of eeprom memory to clear
4	length	In	Number of bytes to clear

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.

See also: ZCBciEepromSize, ZCBciEepromCRC, ZCBciWriteEeprom, ZCBciReadEeprom

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.4 ZCBciCloseAllFiles

Close all open card files on BasicCard.

Syntax:

ZCBCIRET ZCBCILINK ZCBciCloseAllFiles (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciOpenFile, ZCBciCloseFile

Supported cards: ZC2.X, ZC3.X

3.2.2.5 ZCBciCloseFile

Close open card file on BasicCard.

Syntax:

ZCBCIRET ZCBCILINK ZCBciCloseFile (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum);

Parameter:

3. Reference

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
7	FileNum	In	File number of file to close.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciCloseAllFiles, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.6 ZCBciCurDir

Query current directory on card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciCurDir (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath, BYTE cbPath);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	Out	Current directory on card.
5	cbPath	In	Size of buffer pszPath

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_OVERFLOW	Directory name exceeds buffer size

See also: ZCBciChDir

Supported cards: ZC2.X, ZC3.X

3.2.2.7 ZCBciDetach

Detach BasicCard from reader device. Card handle cannot be used after call to this function.

Syntax:

ZCBCIRET ZCBCILINK ZCBciDetach (ZCBCICARD card);

Parameter:

Number	Name	Direction	Description
1	Card	in	Handle to card

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCBciAttach

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.8 ZCBciDirCount

Count files and directories on card matching file specification.

Syntax:

ZCBCIRET ZCBCILINK ZCBciDirCount (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszFileSpec, WORD *pCount);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszFileSpec	In	File specification to search for. For example “\\Test*”
5	*pCount	Out	Number of files or directories match given specification

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciDirFile

Supported cards: ZC2.X, ZC3.X

3.2.2.9 ZCBciDirFile

Return file or directory name of card file or directory matching given file specification and number. ZCBciDirCount must be called before with same file specification.

Syntax:

3. Reference

ZCBCIRET ZCBCILINK ZCBciDirFile(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszFileSpec, WORD num, CHAR *pszFile, BYTE cbFile);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszFileSpec	In	File specification to search for. For example "\\Test\\"*
5	num	In	Number of files or directory to return
6	*pszFile	Out	Name of matched directory or file
7	cbFile	In	Size of buffer pszFile

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_OVERFLOW	File name exceeds buffer size

See also: ZCBciDirCount

Supported cards: ZC2.X, ZC3.X

3.2.2.10 ZCBciEcho

BasicCard ECHO command (Cla C0, Ins 04). Echo data (for testing).

Syntax:

ZCBCIRET ZCBCILINK ZCBciEcho (ZCBCICARD card, WORD *pSW1SW2, BYTE inc, CHAR *pszEcho)

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	inc	In	Increment value to add to every byte in pszEcho by BasicCard echo command
4	*pszEcho	In	Zero terminated string to echo
		Out	Echo string from card

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data

ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.11 ZCBciEepromCRC

BasicCard EEPROM CRC command (Cla C0, Ins 0A). Calculate and return eeprom crc.

Note: Calling ZCBciEepromCRC on an empty (no program loaded) Enhanced BasicCard will destroy the card.

Note: Keep in mind EEPROM CRC will not send WTX, so a call of this command with a big length value may cause a timeout. There may be an enhanced version of ZCBciEepromCRC, which will split this type of call to commands use less length values, in future.

Syntax:

ZCBCIRET ZCBCILINK ZCBciEepromCRC (ZCBCICARD card, WORD *pSW1SW2, WORD start, WORD length, WORD *pCRC);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	start	In	Eeprom address to start calculate
4	length	In	Number of bytes to calculate CRC on
5	*pCRC	Out	Calculated CRC

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.12 ZCBciEepromSize

BasicCard EEPROM SIZE command (Cla C0, Ins 02). Return start and length of card eeprom.

Syntax:

ZCBCIRET ZCBCILINK ZCBciEepromSize (ZCBCICARD card, WORD *pSW1SW2, WORD *pStart, WORD *pLength);

Parameter:

Number	Name	Direction	Description
--------	------	-----------	-------------

3. Reference

1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pStart	Out	Eeprom start address
4	*pLength	Out	Length of BasicCard eeprom

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.13 ZCBciEndEncryption

BasicCard END ENCRYPTION command (Cla C0, Ins 12). Stop transparent encryption.

Syntax:

ZCBCIRET ZCBCILINK ZCBciEndEncryption (ZCBCICARD card, WORD *pSW1SW2);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: ZCBciStartEncryption

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.14 ZCBciEraseFile

Erase card file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciEraseFile (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	File to remove

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.15 ZCBciFileGet

Get content of open file at current file position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileGet (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	Out	Data read from file
6	cbData	In	Number of data bytes requested

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_MISSING_PARM	Not enough data received

3. Reference

See also: ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.16 ZCBciFileGetPos

Get content of open file at specified file position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileGetPos (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, DWORD pos, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	Pos	In	Requested read position
6	*pData	Out	Data read from file
7	cbData	In	Number of data bytes requested

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARAM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_MISSING_PARAM	Not enough data received

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.17 ZCBciFileLength

Return file length of open file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileLength (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, DWORD *pFileLength);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file to query

5	*pFileLength	Out	Length of file (in byte)
---	--------------	-----	--------------------------

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciQueryEof

Supported cards: ZC2.X, ZC3.X

3.2.2.18 ZCBciFilePrint

Print string at current position to file. This function just calls ZCBciFileWrite.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFilePrint (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, CHAR *pszVal);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pszVal	In	Zero terminated string to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.19 ZCBciFilePut

Put content of pData to open file at current file position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFilePut (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

3. Reference

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	In	Data write to file
6	cbData	In	Number of data bytes to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also:

Supported cards: ZC2.X, ZC3.X

3.2.2.20 ZCBciFilePutPos

Put content of pData to open file at specified file position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFilePutPos (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, DWORD pos, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	pos	In	Requested write position
6	*pData	In	Data write to file
7	cbData	In	Number of data bytes to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.21 ZCBciFileRead

Read the content of file at current position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileRead (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	Out	Data read from file
6	cbData	In	Number of data bytes requested

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.22 ZCBciFileReadLine

Read line from file at current position to pszLine

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadLine (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, CHAR *pszLine, BYTE cbLine);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pszLine	Out	Data read from file
6	cbLine	In	Size of buffer pszLine

Return values:

3. Reference

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_OVERFLOW	Data exceeds buffer size

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.23 ZCBciFileReadLong

Read long integer value from file at current position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadLong (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, LONG *plVar);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*plVar	Out	Long value read from file

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.24 ZCBciFileReadSingle

Read float (ZCBasic single) value from file at current position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadSingle (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, float *plVar);

Parameter:

Number	Name	Direction	Description
--------	------	-----------	-------------

1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pflVar	Out	Float value read from file

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.25 ZCBciFileReadString

Read a string from a file at current position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadString (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, CHAR *pszVar, BYTE cbVar);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pszVar	Out	String value read from file
6	cbVar	In	Size of buffer pszVar

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_OVERFLOW	String exceeds buffer size

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3. Reference

3.2.2.26 ZCBciFileReadUser

Read user type data from a file at current position to pData.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadUser (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE *pcbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	Out	User data read from file
6	*pcbData	In	Size of buffer pData
		Out	Number of bytes read

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_OVERFLOW	Data exceeds buffer size

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.27 ZCBciFileReadUser2

Read block from a file at current position.

Note: In opposition to ZCBciFileReadUser cbData must match length of data object stored in file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileReadUser2 (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	Out	User data read from file
6	cbData	In	Number of bytes requested

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFileGet, ZCBciFileGetPos, ZCBciFileRead, ZCBciFileReadLine, ZCBciFileReadLong, ZCBciFileReadSingle, ZCBciFileReadString, ZCBciFileReadUser, ZCBciFileReadUser2, ZCBciOpenFile

Supported cards: ZC2.X, ZC3.X

3.2.2.28 ZCBciFileWrite

Write data at current position to file. In difference to ZCBciFileWriteUser data is not prefixed with type byte, before written to file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileWrite (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	In	Data to write
6	cbData	In	Number of bytes to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.29 ZCBciFileWriteLong

Write long value at current position to file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileWriteLong (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, LONG lVal);

Parameter:

3. Reference

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	lVal	In	Long value to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.30 ZCBciFileWriteSingle

Write float (ZCBasic single) value at current position to file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileWriteSingle (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, float flVal);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	flVal	In	Float value to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.31 ZCBciFileWriteString

Write string data at current position to file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileWriteString (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, CHAR *pszVal);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pszVal	In	Zero terminated string to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.32 ZCBciFileWriteUser

Write user type data at current position to file.

Syntax:

ZCBCIRET ZCBCILINK ZCBciFileWriteUser (ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file
5	*pData	In	Data to write
6	cbData	In	Number of bytes to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful

3. Reference

ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFilePut, ZCBciFilePutPos, ZCBciFileWrite, ZCBciFileWriteLong, ZCBciFileWriteSingle, ZCBciFileWriteString, ZCBciFileWriteUser, ZCBciFilePrint

Supported cards: ZC2.X, ZC3.X

3.2.2.33 ZCBciGetApplicationID

BasicCard GET APPLICATION ID command (Cla C0, Ins 0E). Get application id.

Syntax:

ZCBCIRET ZCBCILINK ZCBciGetApplicationID (ZCBCICARD card, WORD *pSW1SW2, CHAR *pszID, BYTE cbID);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pszID	Out	Application ID
4	cbID	In	Size of buffer pszID
5	*pCRC	Out	Calculated CRC

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.34 ZCBciGetAttr

Get file or directory attributes.

Syntax:

ZCBCIRET ZCBCILINK ZCBciGetAttr(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath, WORD *pAttrib);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word

3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	File or directory to query
5	*pAttrib	Out	Returned attributes (see fa... for details on attributes)

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: 3.2.4 Constants

Supported cards: ZC2.X, ZC3.X

3.2.2.35 ZCBciGetFilepos

Get current file pointer position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciGetFilepos(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, DWORD *pFilePos);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file to query
5	*pFilePos	Out	Current file position

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciSetFilepos

Supported cards: ZC2.X, ZC3.X

3.2.2.36 ZCBciGetLockInfo

Get lock information of file or directory.

Syntax:

ZCBCIRET ZCBCILINK ZCBciGetLockInfo(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath, PZCBCILOCKINFO pInfo);

3. Reference

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	File or directory to query
5	*pInfo	Out	Lock information of file or directory

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
ZCBCI_ERROR_MISSING_PARM	Not enough data received

See also: ZCBCILOCKINFO, PZCBCILOCKINFO

Supported cards: ZC2.X, ZC3.X

3.2.2.37 ZCBciGetState

BasicCard GET STATE command (Cla C0, Ins 00) return card state (ZCBCI_STATE_NEW, ZCBCI_STATE_LOAD, ZCBCI_STATE_TEST or ZCBCI_STATE_RUN)

Syntax:

ZCBCIRET ZCBCILINK ZCBciGetState(ZCBCICARD card, WORD *pSW1SW2, BYTE *pState, WORD *pVersion);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pState	Out	Card state. Either ZCBCI_STATE_NEW, ZCBCI_STATE_LOAD, ZCBCI_STATE_TEST or ZCBCI_STATE_RUN
4	*pVersion	Out	BasicCard version for example 0x203 if ZC2.3

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: ZCBciSetState

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.38 ZCBciMkDir

Make directory on card.

Syntax:

```
ZCBCIRET ZCBCILINK ZCBciMkDir(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath);
```

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	Directory name to create. For example "\\Test\\NewDir"

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARAM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciRmdir

Supported cards: ZC2.X, ZC3.X

3.2.2.39 ZCBciOpenFile

Open file on BasicCard.

Note: There is a difference in using ZCBCiFileWriteUser and ZCBciFilePut or ZCBciFilePutPos. Write functions do not result in same binary file data as done by put functions. You must use put/get functions when file open mode is binary (ZCBCI_MODE_BINARY) or random (ZCBCI_MODE_RANDOM), and you must use read/write functions otherwise. See BasicCard reference manual for more details.

Syntax:

```
ZCBCIRET ZCBCILINK ZCBciOpenFile(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath, BYTE mode, WORD recordlen, WORD *pFileNum);
```

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	File to open
5	mode	In	Any valid combination of ZCBCI_MODE_...

3. Reference

			ZCBCI_ACCESS_.. and ZCBCI_SHARE_.. use ' ' to concatenate values. Note: Do not concatenate ZCBCI_SHARE_LOCKREAD and ZCBCI_SHARE_LOCKWRITE by ' '. Use ZCBCI_SHARE_LOCKREADWRITE instead.
6	recordlen	In	Record/block size used by file
7	*pFileNum	Out	File number required to access open file.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciCloseFile

Supported cards: ZC2.X, ZC3.X

3.2.2.40 ZCBciQueryEof

Query if end of file is reached.

Syntax:

ZCBCIRET ZCBCILINK ZCBciQueryEof(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, BOOL *pfEof);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file to query
5	*pfEof	Out	TRUE if end of file reached, otherwise FALSE

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciFileLength

Supported cards: ZC2.X, ZC3.X

3.2.2.41 ZCBciReadEeprom

BasicCard READ EEPROM command (Cla C0, Ins 08). Read cbData bytes from eeprom starting at address start.

Syntax:

ZCBCIRET ZCBCILINK ZCBciReadEeprom(ZCBCICARD card, WORD *pSW1SW2, WORD start, BYTE *pData, BYTE cbData);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	start	In	Eeprom address to start read
4	*pData	Out	Data read from eeprom
5	cbData	In	Number of bytes to read

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameter is missing

See also: ZCBciWriteEeprom, ZCBciClearEeprom, ZCBciEepromCRC, ZCBciEepromSize

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.42 ZCBciReadKeyFile

Read ZCBasic key file as generated by “keygen”.

Syntax:

ZCBCIRET ZCBCILINK ZCBciReadKeyFile(ZCBCICARD card, CHAR *pszFile);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pszFile	In	Name of keyfile (for example “c:\\BasicCrd\\MyProj\\keys.bas”)

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_INVALID_FILE	Invalid key file

3. Reference

See also: ZCBciSetKey, ZCBciSetPoly

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.43 ZCBciRename

Rename or move a file or directory.

Syntax:

ZCBCIRET ZCBCILINK ZCBciRename(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszOldPath, CHAR *pszNewPath);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszOldPath	In	Directory or file to rename or move.
5	*pszNewPath	In	New directory or file name

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also:

Supported cards: ZC2.X, ZC3.X

3.2.2.44 ZCBciRmdir

Remove directory from card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciRmdir(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath)

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	Directory name to remove.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful

ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciMkDir

Supported cards: ZC2.X, ZC3.X

3.2.2.45 ZCBciSetFileLock

Set lock information of file or directory.

Syntax:

ZCBCIRET ZCBCILINK ZCBciSetFileLock(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, CHAR *pszPath, ZCBCILOCK type, BOOL fLock, BYTE KeyCount, BYTE Key1, BYTE Key2);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	*pszPath	In	File or directory to set
5	type	In	Lock type (ZCBCI_LOCK_..)
6	fLock	In	TRUE lock file, FALSE unlock file
7	KeyCount	In	KeyCount number of access keys (0..2)
8	Key1	In	Number of first key (ignored if KeyCount<1)
9	Key2	In	Number of second key (ignored if KeyCount<2)

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: 3.2.4 Constants

Supported cards: ZC2.X, ZC3.X

3.2.2.46 ZCBciSetFilepos

Set current file pointer position.

Syntax:

ZCBCIRET ZCBCILINK ZCBciSetFilepos(ZCBCICARD card, WORD *pSW1SW2, BYTE *pFileErr, WORD FileNum, DWORD FilePos);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card

3. Reference

2	*pSW1SW2	Out	Command status word
3	*pFileErr	Out	File error code (fe..) as returned by card
4	FileNum	In	Number of file to query
5	FilePos	In	New file position

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)

See also: ZCBciGetFilepos

Supported cards: ZC2.X, ZC3.X

3.2.2.47 ZCBciSetKey

Set encryption key. Key set by this function must match key declared in card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciSetKey(ZCBCICARD card, BYTE keynum, BYTE *pKey, BYTE cbKey);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	Keynum	In	Number of key to set
3	*pKey	In	Key
4	cbKey	In	Size of key. Either 8 or 16.

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCBciSetPoly, ZCBciReadKeyFile

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.48 ZCBciSetPoly

Set polynomial for encryption. Polynomial set by this function must match the one declared in card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciSetPoly(ZCBCICARD card, DWORD polynomA, DWORD polynomS);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card

2	polynomA	In	Polynomial A
3	polynomS	In	Polynomial S

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

See also: ZCBciSetKey, ZCBciReadKeyFile

Supported cards: ZC1.X, (ZC2.X), (ZC3.X)

3.2.2.49 ZCBciSetState

BasicCard SET STATE command (Cla C0, Ins 0C). Set card state (ZCBCI_STATE_LOAD, ZCBCI_STATE_TEST or ZCBCI_STATE_RUN)

Note: You need to reset card (ZCCriReconnect) before doing any further action on this card.

Syntax:

ZCBCIRET ZCBCILINK ZCBciSetState(ZCBCICARD card, WORD *pSW1SW2, BYTE state);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	state	In	State to set. Either ZCBCI_STATE_LOAD, ZCBCI_STATE_TEST or ZCBCI_STATE_RUN)

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.

See also: ZCBciGetState

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.50 ZCBciStartEncryption

BasicCard START ENCRYPTION command (Cla C0, Ins 10). Start transparent encryption. Since encryption is transparently done by BasicCard-C-API ZCBCI you must have done a proper key setup (ZCBciSetKey, ZCBciSetPoly, ZCBciReadKeyFile) before calling this.

Syntax:

ZCBCIRET ZCBCILINK ZCBciStartEncryption(ZCBCICARD card, WORD *pSW1SW2, BYTE algo, BYTE key, DWORD random);

Parameter:

3. Reference

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	algo	In	Encryption algorithm to use. Either ZCBCI_SG_LFSR, ZCBCI_SG_LFSR_CRC, ZCBCI_DES or ZCBCI_3DES
4	key	In	Number of encryption key to use
5	random	In	Random number used in conjunction with card random number as initial vector for encryption

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameter is missing

See also: ZCBciEndEncryption

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.51 ZCBciTransaction

Do transaction or in other words call a card command.

Syntax:

ZCBCIRET ZCBCILINK ZCBciTransaction(ZCBCICARD card, BYTE Cla, BYTE Ins, PZCBCIPARM pData, BYTE bDataCount, WORD *pSW1SW2);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	Cla	In	Command class byte (first of 2 byte pair given in ZCBasic Declare Command source line)
3	Ins	In	Command instruction byte (second of 2 byte pair given in ZCBasic Declare Command source line)
4	*pData	In	Input parameter for card command
		Out	Output parameter from card command
5	bDataCount	In	Number of command parameters
6	*pSW1SW2	Out	Command status word

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call

ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameter is missing

See also: 2.4.4 Call User Defined Card Command by Use of ZCBCI, ZCBciTransaction2

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.2.2.52 ZCBciTransaction2

Do enhanced transaction.

Syntax:

ZCBCIRET ZCBCILINK ZCBciTransaction2(ZCBCICARD card, BYTE Cla, BYTE Ins, BYTE P1, BYTE P2, WORD Lc, WORD Le, PZCBCIPARM pData, BYTE DataCount, WORD *pSW1SW2);

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	Cla	In	Command class byte (first of 2 byte pair given in ZCBasic Declare Command source line)
3	Ins	In	Command instruction byte (second of 2 byte pair given in ZCBasic Declare Command source line)
4	P1	In	Parameter 1
5	P2	In	Parameter 2
6	Lc	In	Lc (0..255) or ZCBCI_DEFAULT_LC
7	Le	In	Le (0..255) ZCBCI_DISABLE_LE or ZCBCI_DEFAULT_LE
8	*pData	In	Input parameter for card command
		Out	Output parameter from card command
9	bDataCount	In	Number of command parameters
10	*pSW1SW2	Out	Command status word

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.
ZCBCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: 2.4.4 Call User Defined Card Command by Use of ZCBCI, ZCBciTransaction

Supported cards: ZC1.X, ZC2.X, ZC3.X

3. Reference

3.2.2.53 ZCBciWriteEeprom

BasicCard WRITE EEPROM command (Cla C0, Ins 06). Write cbData bytes to eeprom starting at address start.

Syntax:

```
ZCBCIRET ZCBCILINK ZCBciWriteEeprom(ZCBCICARD card, WORD *pSW1SW2, WORD start, BYTE *pData, BYTE cbData);
```

Parameter:

Number	Name	Direction	Description
1	Card	In	Handle to card
2	*pSW1SW2	Out	Command status word
3	start	In	Start address of eeprom memory to write
4	*pData	In	Data to write to eeprom
5	length	In	Number of bytes to write

Return values:

Value	Description
ZCCRI_NOERROR	No error, function call successful
ZCCRI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_OVERFLOW	Too much parameter data
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX.

See also: ZCBciClearEeprom, ZCBciReadEeprom

Supported cards: ZC1.X, ZC2.X, (ZC3.X)

3.2.3 Structures and Types

Pointer types to defined types can be constructed by prefix type with 'P'. For example PZCBCIIRET is a pointer to ZCBCIRET. All types are aligned on byte boundaries (#pragma pack(1)).

3.2.3.1 ZCBCIRET, PZCBCIRET

32 bit unsigned ZCBCI-API return value. See "Table 3 ZCBCI return values" for possible values.

C typedef:

```
typedef DWORD ZCBCIRET;  
typedef ZCBCIRET *PZCBCIRET;
```

3.2.3.2 ZCBCIHANDLE, PZCBCIHANDLE

32 bit handle.

C typedef:

```
typedef DWORD ZCBCIHANDLE;  
typedef ZCBCIHANDLE *PZCBCIHANDLE;
```

3.2.3.3 ZCBCICARD, PZCBCICARD

Handle to BasicCard context.

C typedef:

```
typedef ZCBCIHANDLE ZCBCICARD;
```

```
typedef ZCBCICARD *PZCBCICARD;
```

3.2.3.4 ZCBCIPARMTYPE

Enumeration to set used parameter type. Following values are defined:

Value	Description
ZCBCIBYTE	8 bit byte value parameter
ZCBCIINT	16 bit integer value parameter
ZCBCILONG	32 bit long value parameter
ZCBCISTRING	Zero terminated string parameter. Must be last parameter if used.
ZCBCISTRINGN	Zero terminated string parameter to be passed as ZCBasic String*N
ZCBCIUSER	User type parameter
ZCBCIRAW	Raw data. Must be the one and only parameter. Not required for normal use.
ZCBCIOCXSTRING	Used by OCX to pass Strings. Should not be used from C/C++ programs.
ZCBCINULL	Empty parameter used by OCX. Should not be used from C/C++ programs.
ZCBCISINGLE	Floating point number parameter

C typedef:

```
typedef enum _ZCBCIPARMTYPE
```

```
{
```

```
    ZCBCIBYTE,
```

```
    ZCBCIINT,
```

```
    ZCBCILONG,
```

```
    ZCBCISTRING,
```

```
    ZCBCISTRINGN,
```

```
    ZCBCIUSER,
```

```
    ZCBCIRAW,
```

```
    ZCBCIOCXSTRING,
```

```
    ZCBCINULL,
```

```
    ZCBCISINGLE,
```

```
} ZCBCIPARMTYPE;
```

3.2.3.5 ZCBCIPARMBYTE, PZCBCIPARMBYTE

8 bit byte type parameter. Enumeration type ZCBCIBYTE.

C typedef:

```
typedef BYTE ZCBCIPARMBYTE;
```

```
typedef ZCBCIPARMBYTE *PZCBCIPARMBYTE;
```

3. Reference

3.2.3.6 ZCBCIPARMINT, PZCBCIPARMINT

16 bit signed integer type parameter. Enumeration type ZCBCIINT.

C typedef:

```
typedef INT ZCBCIPARMINT;  
typedef ZCBCIPARMINT *PZCBCIPARMINT;
```

3.2.3.7 ZCBCIPARMLONG, PZCBCIPARMLONG

32 bit signed long integer type parameter. Enumeration type ZCBCILONG.

C typedef:

```
typedef LONG ZCBCIPARMLONG;  
typedef ZCBCIPARMLONG *PZCBCIPARMLONG;
```

3.2.3.8 ZCBCIPARMSINGLE, PZCBCIPARMSINGLE

Floating point number type parameter. Enumeration type ZCBCISINGLE.

C typedef:

```
typedef FLOAT ZCBCIPARMSINGLE;  
typedef ZCBCIPARMSINGLE *PZCBCIPARMSINGLE;
```

3.2.3.9 ZCBCIPARMSTRING, PZCBCIPARMSTRING

Zero terminated string type parameter. Enumeration type ZCBCISTRING.

Component	Description
cbString	Size of buffer pString
pString	Pointer to zero terminated string.

C typedef:

```
typedef struct _ZCBCIPARMSTRING  
{  
    BYTE cbString;  
    CHAR *pString;  
} ZCBCIPARMSTRING;  
typedef ZCBCIPARMSTRING *PZCBCIPARMSTRING;
```

3.2.3.10 ZCBCIPARMSTRINGN, PZCBCIPARMSTRINGN

Zero terminated string type parameter to be passed as ZCBasic type STRING*N. Enumeration type ZCBCISTRINGN

Component	Description
bLength	Length of string (equals to N)
pString	Pointer to zero terminated string.

C typedef:

```
typedef struct _ZCBCIPARMSTRINGN  
{  
    BYTE bLength;  
    CHAR *pString;
```

```

} ZCBCIPARMSTRINGN;
typedef ZCBCIPARMSTRINGN *PZCBCIPARMSTRINGN;

```

3.2.3.11 ZCBCIPARMUSER, PZCBCIPARMUSER

User type parameter. Enumeration type ZCBCIUSER.

Component	Description
bSize	Size of user data
pData	Pointer to data

C typedef:

```

typedef struct _ZCBCIPARMUSER
{
    BYTE bSize;
    BYTE *pData;
} ZCBCIPARMUSER;
typedef ZCBCIPARMUSER *PZCBCIPARMUSER;

```

3.2.3.12 ZCBCIPARMRAW, PZCBCIPARMRAW

Raw data parameter. Enumeration type ZCBCIRAW. Not needed for normal use.

Component	Description
bSize	Size of buffer pData
bLen	Length of data to sent on input, length of data received on output.
pData	Pointer to data

C typedef:

```

typedef struct _ZCBCIPARMRAW
{
    BYTE bSize;
    BYTE bLen;
    BYTE *pData;
} ZCBCIPARMRAW;
typedef ZCBCIPARMRAW *PZCBCIPARMRAW;

```

3.2.3.13 ZCBCIPARMDUMMY;

Place holder, do not use.

C typedef:

```

typedef BYTE ZCBCIPARMDUMMY[12];

```

3.2.3.14 ZCBCIPARMDATA

Union to hold data part of transaction parameter.

Component	Description
valbyte	Access type for 8 bit byte. ZCBCIBYTE
valint	Access type for 16 bit integer. ZCBCIINT

3. Reference

vallong	Access type for 32 bit long. ZCBCILONG
valstring	Access type for string value. ZCBCISTRING
valstringn	Access type for string(n) value. ZCBCISTRINGN
valuser	Access type for user type value. ZBCIUSER
valraw	Access type for raw data. ZCBCIRAW
donotuse	Place holder to reserve memory for later use.

C typedef:

```
typedef union _ZCBCIPARMDATA
{
    ZCBCIPARMBYTE  valbyte;
    ZCBCIPARMINT   valint;
    ZCBCIPARMLONG  vallong;
    ZCBCIPARMSTRING valstring;
    ZCBCIPARMSTRINGN valstringn;
    ZCBCIPARMUSER  valuser;
    ZCBCIPARMRAW   valraw;
    ZCBCIPARMOCXSTRING valocxstring;
    ZCBCIPARMDDUMMY donotuse;
    ZCBCIPARMSINGLE valsingle;
} ZCBCIPARMDATA;
```

3.2.3.15 ZCBCIPARM, PZCBCIPARM

Complete transaction parameter.

Component	Description
parmtype	Specify type of given parameter
data	Parameter data

C typedef:

```
typedef struct _ZCBCIPARM
{
    ZCBCIPARMTYPE parmtype;
    ZCBCIPARMDATA data;
} ZCBCIPARM;
typedef ZCBCIPARM *PZCBCIPARM;
```

3.2.3.16 ZCBCILOCKINFO, PZCBCILOCKINFO

File directory lock info.

Component	Description
ReadLock	Read lock value. Either liAllowed, liKeyed1 (encryption with key ReadKey[0] required), liKeyed2 (encryption with key ReadKey[1] required) or

	liForbidden
WriteLock	Write lock value. Either liAllowed, liKeyed1 (encryption with key WriteKey[0] required), liKeyed2 (encryption with key WriteKey[1] required) or liForbidden
CustomLock	Custom lock value. Either liAllowed, liUnlocked or liLocked.
ReadKey	Read lock keys (key number). If ReadLock is either liKeyed1 or liKeyed2.
WriteKey	Write lock keys (key number). If WriteLock is either liKeyed1 or liKeyed2.

C typedef:

```
typedef struct _ZBCILOCKINFO
{
    BYTE ReadLock;
    BYTE WriteLock;
    BYTE CustomLock;
    BYTE ReadKey[2];
    BYTE WriteKey[2];
} ZBCILOCKINFO;
typedef ZBCILOCKINFO *PZBCILOCKINFO;
```

3.2.3.17 ZBCILOCK, PZBCILOCK

Enumeration type to modify file or directory locks.

Value	Description
ZBCI_LOCK_CUSTOM	Set custom lock
ZBCI_LOCK_READ	Set read lock
ZBCI_LOCK_WRITE	Set write lock
ZBCI_LOCK_READWRITE	Set read and write lock

C typedef:

```
typedef enum _ZBCILOCK
{
    ZBCI_LOCK_CUSTOM=0x0,
    ZBCI_LOCK_READ =0x4,
    ZBCI_LOCK_WRITE =0x8,
    ZBCI_LOCK_READWRITE=0xC,
} ZBCILOCK;
typedef ZBCILOCK *PZBCILOCK;
```

3.2.4 Constants

For status word codes (sw..) and file error codes (fe..) refer to BasicCard manual.

Value (Hex)	Symbolic name	Description
0	ZBCI_NOERROR	Function call successful, no error

3. Reference

0x02000000	ZCBCI_ERROR_PARM	Invalid parameter passed to function call
0x02000001	ZCBCI_ERROR_MISC	Misc. error
0x02000002	ZCBCI_ERROR_MEM	Out of memory
0x02000003	ZCBCI_ERROR_OVERFLOW	Buffer overflow
0x02000004	ZCBCI_ERROR_TRANS_FAILED	Transaction failed. *pSW1SW2 differs from 0x9000 or 0x61XX or file error occurred (*pFileErr!=0)
0x02000005	ZCBCI_ERROR_MISSING_PARM	Not enough return data received to fill return variables.
0x02000006	ZCBCI_ERROR_INVALID_FILE	Invalid file specified. File does not exist or is not accessible

Table 3 ZCBCI return values

Symbolic name	Value	Description
ZCBCILINK	APIENTRY	ZCBCI-API linkage convention
ZCBCI_STATE_NEW	0	New state of BasicCard
ZCBCI_STATE_LOAD	1	Load state of BasicCard
ZCBCI_STATE_TEST	2	Test state of BasicCard
ZCBCI_STATE_RUN	3	Run state of BasicCard
ZCBCI_SG_LFSR	0x11	Encryption algorithm Shrinking Generator – Linear Feedback Shift Register
ZCBCI_SG_LFSR_CRC	0x12	Encryption algorithm same as above but with CRC
ZCBCI_DES	0x21	Encryption algorithm DES (Data Encryption Standard)
ZCBCI_3DES	0x22	Encryption algorithm Triple-DES
ZCBCI_DEFAULT_LC	0xFFFF	Use default (calculated Lc)
ZCBCI_DEFAULT_LE	0xFFFF	Use default (calculated Le)
ZCBCI_DISABLE_LE	0xFFFE	Do not transmit Le
ZCBCI_MODE_APPEND	0x60	Open file in append mode
ZCBCI_MODE_BINARY	0x20	Open file in binary mode
ZCBCI_MODE_INPUT	0x30	Open file in input mode
ZCBCI_MODE_OUTPUT	0x40	Open file in output mode
ZCBCI_MODE_RANDOM	0x50	Open file in random mode
ZCBCI_ACCESS_READ	0x04	Open file for read access
ZCBCI_ACCESS_WRITE	0x08	Open file for write access
ZCBCI_ACCESS_READWRITE	0x0C	Open file for read and write access
ZCBCI_SHARE_SHARED	0x03	Open file in shared mode
ZCBCI_SHARE_LOCKREAD	0x02	Open file with read lock
ZCBCI_SHARE_LOCKWRITE	0x01	Open file with write lock
ZCBCI_SHARE_LOCKREADWRITE	0x00	Open file with read and write lock
faDirectory	0x0010	File attribute directory

faCardFile	0x0040	File attribute card file
liAllowed	0	File lock allowed
liKeyed1	1	File lock keyed 1
liKeyed2	2	File lock keyed 2
liForbidden	3	File lock forbidden
liUnlocked	1	File lock unlocked (custom lock)
liLocked	2	File lock locked (custom lock)

Table 4 Other ZCBCI constants

3. Reference

4. Index

- D**
- DefReg 2
 - DEFREG.BAT 3
- F**
- faCardFile 59
 - faDirectory 58
- L**
- liAllowed 59
 - liForbidden 59
 - liKeyed1 59
 - liKeyed2 59
 - liLocked 59
 - liUnlocked 59
- P**
- PZCBCICARD 52
 - PZCBCIHANDLE 52
 - PZCBCILOCK 57
 - PZCBCILOCKINFO 56
 - PZCBCIPARM 56
 - PZCBCIPARMBYTE 53
 - PZCBCIPARMINT 54
 - PZCBCIPARMLONG 54
 - PZCBCIPARMRAW 55
 - PZCBCIPARMSINGLE 54
 - PZCBCIPARMSTRING 54
 - PZCBCIPARMSTRINGN 54
 - PZCBCIPARMUSER 55
 - PZCBCIRET 52
 - PZCCRIATR 18
 - PZCCR IHANDLE 18
 - PZCCRIREADER 18
 - PZCCRIREADERNAME 18
 - PZCCRIRET 18
 - PZCCRISERVICENAME 18
 - PZCCRIVER 18
- T**
- TestCri 2
 - TESTCRI.EXE 3
- V**
- valbyte 55
 - valint 55
 - vallong 56
 - valraw 56
 - valstring 56
 - valstringn 56
 - valuser 56
- Z**
- Z CBCI_MODE_APPEND 58
 - ZCBCI 2, 5
 - ZCBCI.DLL 3
 - ZCBCI.H 4
 - ZCBCI_3DES 58
 - ZCBCI_ACCESS_READ 58
 - ZCBCI_ACCESS_READWRITE 58
 - ZCBCI_ACCESS_WRITE 58
 - ZCBCI_DEFAULT_LC 58
 - ZCBCI_DEFAULT_LE 58
 - ZCBCI_DES 58
 - ZCBCI_DISABLE_LE 58
 - ZCBCI_ERROR_INVALID_FILE 58
 - ZCBCI_ERROR_MEM 58
 - ZCBCI_ERROR_MISC 58
 - ZCBCI_ERROR_MISSING_PARM 58
 - ZCBCI_ERROR_OVERFLOW 58
 - ZCBCI_ERROR_PARM 58
 - ZCBCI_ERROR_TRANS_FAILED 6, 58
 - ZCBCI_MODE_BINARY 58
 - ZCBCI_MODE_INPUT 58
 - ZCBCI_MODE_OUTPUT 58
 - ZCBCI_MODE_RANDOM 58
 - ZCBCI_NOERROR 57
 - ZCBCI_SG_LFSR 58
 - ZCBCI_SG_LFSR_CRC 58
 - ZCBCI_SHARE_LOCKREAD 58
 - ZCBCI_SHARE_LOCKREADWRITE 58
 - ZCBCI_SHARE_LOCKWRITE 58
 - ZCBCI_SHARE_SHARED 58
 - ZCBCI_STATE_LOAD 58
 - ZCBCI_STATE_NEW 58
 - ZCBCI_STATE_RUN 58
 - ZCBCI_STATE_TEST 58
 - ZCBciAttach 5, 21
 - ZCBCIBYTE 6, 53
 - ZCBCICARD 52
 - ZCBciChDir 22
 - ZCBciClearEeprom 22
 - ZCBciCloseAllFiles 23
 - ZCBciCloseFile 23
 - ZCBciCurDir 24
 - ZCBciDetach 5, 24
 - ZCBciDirCount 25
 - ZCBciDirFile 25
 - ZCBciEcho 5, 26
 - ZCBciEepromCRC 27
 - ZCBciEepromSize 27
 - ZCBciEraseFile 28
 - ZCBciFileGet 29
 - ZCBciFileGetPos 30
 - ZCBciFileLength 30
 - ZCBciFilePrint 31
 - ZCBciFilePut 31
 - ZCBciFilePutPos 32
 - ZCBciFileRead 33
 - ZCBciFileReadLine 33
 - ZCBciFileReadLong 34
 - ZCBciFileReadSingle 34
 - ZCBciFileReadString 35
 - ZCBciFileReadUser 36
 - ZCBciFileReadUser2 36
 - ZCBciFileWrite 31, 37
 - ZCBciFileWriteLong 37
 - ZCBciFileWriteSingle 38
 - ZCBciFileWriteString 39
 - ZCBciFileWriteUser 39
 - ZCBciGetApplicationID 40
 - ZCBciGetAttr 40

4. Index

ZCBciGetFilepos 41
ZCBciGetLockInfo 41
ZCBciGetState 42
ZCBCIHANDLE 52
ZCBCIINT 6, 53
ZCBCILINK 58
ZCBCILOCK 57
ZCBCILOCKINFO 42, 56
ZCBCILONG 6, 53
ZCBciMkDir 43
ZCBCINULL 53
ZCBCIOCXSTRING 53
ZCBciOpenFile 43
ZCBCIPARM 56
ZCBCIPARMBYTE 53
ZCBCIPARMDATA 55
ZCBCIPARMDUMMY 55
ZCBCIPARMINT 54
ZCBCIPARMLONG 54
ZCBCIPARMRAW 55
ZCBCIPARMSINGLE 54
ZCBCIPARMSTRING 54
ZCBCIPARMSTRINGN 54
ZCBCIPARMTYPE 53
ZCBCIPARMUSER 55
ZCBciQueryEof 44
ZCBCIRAW 53
ZCBciReadEeprom 45
ZCBciReadKeyFile 45
ZCBciRename 46
ZCBCIRET 5, 6, 52
ZCBciRmdir 46
ZCBciSetFileLock 47
ZCBciSetFilepos 47
ZCBciSetKey 48
ZCBciSetPoly 48
ZCBciSetState 49
ZCBCISINGLE 53
ZCBciStartEncryption 49
ZCBCISTRING 6, 53
ZCBCISTRINGN 53
ZCBciTransaction 6, 16, 50
ZCBciTransaction2 51
ZCBCIUSER 53
ZCBciWriteEeprom 52
ZCCRI 1, 4
ZCCRI.DLL 3
ZCCRI.H 4
ZCCRI_CANCEL 19
ZCCRI_ERROR_INITFAILED 19
ZCCRI_ERROR_MEM 19
ZCCRI_ERROR_MISC 19
ZCCRI_ERROR_NO_CARD 19
ZCCRI_ERROR_NO_DEFAULT 19
ZCCRI_ERROR_NOT_SUPPORTED 19
ZCCRI_ERROR_OVERFLOW 19
ZCCRI_ERROR_PARM 19
ZCCRI_ERROR_PROTOCOL 19
ZCCRI_ERROR_READER_BUSY 19
ZCCRI_ERROR_UNKOWN_READER 19
ZCCRI_FEATURE_CHECKCARD 19
ZCCRI_FEATURE_CHECKCARD_ATR 19
ZCCRI_MAX_ATR 19
ZCCRI_MAX_READER_NAME 19
ZCCRI_MAX_SERVICE_NAME 19
ZCCRI_NOERROR 19
ZCCRI_OPTION_FASTINIT 19
ZCCRI_SEPARATOR 19
ZCCRIATR 18
ZCCriCardInReader 8
ZCCriCloseReader 5, 8
ZCCriConnect 4, 9
ZCCRIDEF.H 4
ZCCriDeregisterService 9
ZCCriDisconnect 5, 9
ZCCriGetDefaultReader 10
ZCCriGetFeatures 10
ZCCriGetOptions 11
ZCCriGetOptions2 11
ZCCriGetReaderCount 12
ZCCriGetReaderName 12
ZCCRIHANDLE 18
ZCCRILINK 19
ZCCriOpenReader 4, 12, 14
ZCCRIPRT.H 4
ZCCRIREADER 18
ZCCRIREADERNAME 18
ZCCriReconnect 5, 13
ZCCriReg 2
ZCCRIREG.EXE 3
ZCCriRegisterService 13
ZCCRIRET 18
ZCCriSelectReaderDialog 14
ZCCRISERVICENAME 18
ZCCriSetDefaultReader 14
ZCCriSetLogFile 14
ZCCriSetOptions 15
ZCCriSetOptions2 15
ZCCriTransaction 16
ZCCriTransaction2 16
ZCCRITYP.H 4
ZCCRIVER 18
ZCCriWaitCard 4, 17
ZCRWCYBR.DLL 3
ZCRWL 1
ZCRWPCSC.DLL 3